AD-A262 959

1

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave blank)

2. REPORT DATE
December 31, 1992

3. REPORT TYPE AND DATES COVERED
Quarterly Technical Report, 7/92 -9/92

4. TITLE AND SUBTITLE
An MCM/Chip Concurrent Engineering Validation
Final Technical Report

5. FUNDING NUMBERS
MDA972-92-C-0022

6. AUTHOR(S)
Dr. Hector Moreno
Shaune Stark, BS

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Microelectronics and Computer Technology Corporation
3500 West Balcones Center Dr.
Austin, TX 78759

8. PERFORMING ORGANIZATION REPORT NUMBER
HVE- 042-93

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
DARPA/DSO, Dr. H. Lee Buchanan
3701 N. Fairfax Dr.
Arlington, VA 22203-1714

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION / AVAILABILITY STATEMENT
No restrictions

12b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)

We report on the MOSTsoftware system, which implements a concurrent physical design environment for Multi-Chip Modules. The system integrates the work of design teams distributed across a network and using different CAD systems. At present the following systems have been integrated: Cadence's Edge 2.1, Cadence's Allegro 6.1, AutoDesk's AutoCad 12.0, and Harris' Finesse. Software links were established allowing data from those systems to be shared through a ROSE (Rensselaer Object Storage Environment) database management developed under the sponsorship of the DICE program. The code was written in C++ and uses various methods to feed the information in and obtain it out of the design systems: IGES for Allegro, SKILL for Edge and dfile for Finesse, while the AutoCad link is a direct one.

The DDR2 (Digital Drop Receiver, version 2) multi-chip module from Harris was entered into the system and routed utilizing a redundant route scheme. The exercise used a concurrent approach, the data defining parts and placement entered through Finesse, the parts modified in Edge, the route done in Allegro and the final merge and verification performed with the Edge tool.

14. SUBJECT TERMS
Concurrent Engineering, Multi-Chip Module, Computer-Aided Design, CAD System Integration

15. NUMBER OF PAGES
33

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT
Unclassified

18. SECURITY CLASSIFICATION OF THIS PAGE
Unclassified

19. SECURITY CLASSIFICATION OF ABSTRACT
Unclassified

20. LIMITATION OF ABSTRACT
SAR

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. Z39-18
298-102

93-06982

32pg

Original contains color plates: All DTIC reproductions will be in black and white

3 4 02 131

MCC High Value Electronics Division

Document No. HVE-042-93(P)

Author(s) Hector Moreno and Shaune Stark

Title An MCM/Chip Concurrent Engineering Validation, Final Technical Report

Abstract This report presents the technical progress of the execution of Phase 1 of a project at MCC which was supported by ARPA. The principal project objective was to create and validate a concurrent engineering environment for the design of multichip modules (MCMs). In Phase 1, the focus was on the physical design of MCMs. The objective was to integrate four different MCM layout systems: Edge, Finesse, Allegro 6.1, and AutoCad.

	Contents		Project		Distribution
	Non proprietary		Enabling Technology		unrestricted
	MCC confidential		Flip Chip		MCC internal only
X	HVE proprietary		Laser Bonding		HVE internal only
	Project proprietary		Low Cost Interconnect		Sustaining members
	Single client proprietary		LCI - QTAI	X	Project members
	Quality		Advanced TAB		Single client use only
	draft		Open Systems - HW		
	external review		Open Systems - SW		**Do not distribute to:**
X	finished		Govt. Open Systems		
	External Publication		RwoH		
			PRI		
		X	Government Contract		**Also distribute to:**
			MSDA		ARPA
			Power Sources		
			Portables		

Position	Name	Signature	Date
author	Hector Moreno	[illegible]	3/30/93
project leader	Hector Moreno	[illegible]	3/30/93
reviewed by	Brad Nelson	[illegible]	3/29/93
projects manager	Don Orr	[illegible]	30 Mar /93
program director	Dennis Herrell	[illegible]	3/29/93

MCC

Microelectronics and Computer Technology Corporation

An MCM/Chip Concurrent Engineering Validation

Final Technical Report

Dr. Hector Moreno
Shaune Stark, B.S.
High Value Electronics Division
Austin, TX 78727
March 30, 1993

Sponsored by
Advanced Research Projects Agency
Defense Sciences Office
ARPA Initiative on Concurrent Engineering
ARPA Order No. 8363/07
Issued by ARPA/CMO under Contract #MDA972-92-C-0022

Trademarks

UNIX is a trademark of AT&T Bell Laboratories.
Allegro is a trademark of Cadence Design Systems.
EDGE is a trademark of Cadence Design Systems.
AutoCad is a trademark of AutoDesk.
Finesse is a trademark of Harris Corporation.
SKILL is a language developed at Cadence Design Systems.
C++ is a language developed at AT&T Bell Laboratories
ROSE and ROSE++ (the C++ version of ROSE) have been developed at Rensselaer Polytechnic.

Table of Contents

Summary

This report presents the technical progress of the execution of Phase 1 of a project at the Microelectronics and Computer Technology Corporation (MCC), supported by the Advanced Research Projects Agency (ARPA).

Objectives:

The principal project objective was to create and validate a concurrent engineering environment for the design of Multi-Chip Modules (MCMs).

On Phase 1, the focus was on physical design of MCMs. The objective in this phase was to integrate four different MCM layout systems: Edge, Finesse, Allegro 6.1 and AutoCad.

Technical Problems:

The principal technical problems addressed were the development of four commercial CAD system interfaces to the ROSE (Rensselaer Object Storage Environment) database manager and the exercise of the developed software to perform an actual MCM design in concurrent mode.

General Methodology:

The first step was to become familiar with the ROSE database management system. The next step was to become familiar with the way in which the commercial CAD systems store their data, which allowed us to define an information model in the EXPRESS language. STEP Tools, Inc. provided an EXPRESS to C++ compiler, which permitted us to use the C++ language to implement the interfaces and the information model that defined and structured the persistent objects managed by ROSE. In addition, special programs were written to interface with the CAD system databases both for input and output. These programs were written utilizing the following languages: SKILL for Cadence Edge, C for AutoCad, Finesse's own text file formats and IGES for Cadence Allegro.

Technical Results:

The exercise of the software outlined above proved successful, as we were able to take previously defined parts and netlist for an MCM from the Finesse system into the Cadence Edge system, where they were modified and then resubmitted to the database. The Allegro system was then used to do the routing. Through an efficient design strategy we were able to implement a redundant routing scheme that improves the probability of having an operational MCM upon manufacture. The final design was merged into the database and verified by the Edge software. The defined and placed parts were also sent to the AutoCad system running on a Sun workstation. The total designer time spent on this task was 4 man days, whereas the same module design by traditional, non-concurrent methods takes one man-month.

Important Findings and Conclusions:

The success that was achieved demonstrates the very practical utility of the tools developed under the DICE initiative, notably the ROSE data management software. It is apparent that the utilized object oriented approach to data storage is appropriate for the MCM design task to which it was applied The tools and interfaces developed have utility and application beyond concurrent engineering: they provide a practical integration of diverse MCM and Printed Circuit Board CAD design systems and frameworks.

Significant Hardware Development:

Not applicable.

Special Comments:

None.

Implications for further research:

The success of this phase makes it possible to contemplate the extension of this work to other areas of MCM design, namely: logical, back-annotation, thermal, mechanical and electrical modeling. The efficient performance of ROSE and its associated tools makes it a good candidate as data manager for the extended picture scheme. However, it will be necessary to make improvements and add features to ROSE, such as version control, which do not exist today.

Other possibilities that have been uncovered by this work lie in the area of commercialization of the system that has been implemented. The commercial value arises because it is possible to reuse parts that have been designed elsewhere, fewer errors creep in because data flows from one system to the other in error-free manner, and one does not have the problem of having expensive CAD software available, as much of the work can be done with relatively inexpensive systems. However, more commercial CAD systems need to be interfaced to make the system useful to more design teams and it must be debugged and strengthened.

1. Project Objectives

1.1. Phase 1

To create a concurrent engineering environment suitable for the physical design of Multi-Chip Modules (MCMs) by integrating several different physical layout Computer-Aided Design (CAD) tools through data sharing.

To validate the use of ROSE as a concurrent engineering data manager [ST].

To validate the concurrent engineering approach to MCM design.

To utilize the concurrent engineering environment to design MCMs.

1.2. Phase 2

Repeat the Phase 1 objectives but extending them individually to each of the following areas:

MCM modeling.

MCM simulation.

Design for testability.

This phase will begin upon funding release.

2. Progress Overview

2.1. Technical progress achieved

The project officially started on May 1992 and ended February 28, 1993. During these months the following work was done:

A study of the ROSE software was made. Two MCC software developers attended an introductory course offered by STEP Tools, Inc. and came back with knowledge and documentation, which was shared with others at MCC. The ROSE software was installed at MCC in Austin TX and at Harris EDA in Rochester NY. The tutorials offered by STEP Tools were studied and understood.

In parallel with these activities MCC undertook a study of the input/output to the targeted CAD systems. It was decided that the best way to interface with the Cadence EDGE design system was through a procedural interface supported by an interpreted language called SKILL, proprietary to Cadence Design Systems. A better, faster and more efficient interface using the C language exists but was not used because of economic reasons.

When the proposal for this project was written *and presented to ARPA (then known as* DARPA) the Allegro system belonged to Valid Systems. Valid merged with Cadence in 1992. MCC had been working with Valid Systems in the definition and implementation of Allegro 6.0, which includes an input/output interface through the IGES language. This interface was the logical choice for our project.

Harris EDA decided to implement an interface to the ROSE database based on a textual description of a Finesse design known as a "Dfile". This was as far as output from the Finesse system was concerned. They implemented the input utilizing a proprietary macro language.

For AutoCad we used a C language procedural interface which is supported by AutoDesk, proprietor of that system. This interface has the advantage that it does not go through any intermediate form like the other systems.

The next step was to study the data being described and to find a common denominator. When this was done, we abstracted the information and created an EXPRESS model for it. Utilizing the STEP Tools , Inc. "express2c++" compiler, the information model was compiled into a set of ROSE C++ classes. Then software was written to implement the interfaces from all of the CAD systems to these classes. The classes were managed with ROSE software libraries and methods. In the following we will refer to our implementation of the object data with these classes as "MOST format". MOST being an acronym for MCC's Open Systems Tool, which is the name we have given to the implemented software system.

After thorough study of the IGES entities that Allegro utilizes, it was decided to interface from an Allegro IGES file describing an MCM to the ROSE classes using the IGES toolkit provided by STEP Tools, Inc. . This tool allows the creation of ROSE class images of the IGES entities, and vice versa, the creation of an IGES file given a set of correctly structured ROSE IGES class objects. It must be stated that the ROSE IGES classes are related to but not identical to the MCC classes. The reason is that for the most part, this project was concerned with two-dimensional design objects, while the IGES classes are general enough to describe three-dimensional objects. In addition, it is not easy to describe things such as electrical connectivity using IGES, while the MCC information model incorporates net and element pin netlist description objects. The IGES toolkit was utilized solely for the geometric information and the main benefit it provided was in allowing us to eliminate the tasks of IGES file creation and parsing

Once the software was implemented, we took the parts and netlist for the Harris DDR2 MCM and converted them from Finesse into MOST format. Then we designed a version of the DDR2 using MCC's QTAI technology design rules [DC91] and implemented a strategy to double the number of nets in order to give the designed module redundancy and enhanced manufacturability. The design was performed in a concurrent mode, with a designer working in the Edge environment and another in the Allegro environment. The concurrent design experiment was successful, and the completed design was achieved in four man days, compared to the usual one man-month that it takes to complete a comparable task. The DDR1 MCM, which had different components and netlist but the same degree of difficulty served as basis for comparison.

Thus, MCC has accomplished everything that was planned for this project phase.

3. Technical Problems

3.1. Areas addressed

During the project we focused exclusively on Phase 1 objectives. The principal technical areas that were addressed were the following:

The definition of the entities necessary for the description of the MCM layout data, taking into account that there are a number of designers, each using one of a variety of heterogeneous design systems who need read/write access. These entities were the building blocks from which the information model was built

The overall architecture of the concurrent engineering environment.

The process and practice through which the actual concurrent design activity took place.

The improvement of the definition of the entities that are necessary for the description of the MCM layout data.

The integration of the Cadence Edge system through its SKILL interface.

The integration of the Allegro CAD system through its IGES input/output interface.

The integration of the Finesse CAD system.

The integration of the AutoCad system.

The implementation of a UNIX server to allow the Concurrent Design team to work as a unit.

The installation and support of the system at Harris Corporation in Melbourne, Florida.

The demonstration of the software at ARPA.

The implementation of a concurrent design experiment utilizing the integrated CAD environment.

4. General Methodology

4.1. Layout Information Model

In order to define the entities needed to represent the MCM layout information it is necessary to understand the information which the individual CAD systems store and which is required and needed for their proper operation. Then one needs to abstract and deal first with that subset which is common to all of the systems and use it as a nucleus or core. From this point one can expand and tailor the core to include what will be required in the concurrent use of the integrated systems.

In our case, there are two basic types of entities:

Geometric information entities.

Connectivity information entities.

Our method was to examine what data types were required by a system which is predominantly used for layout, which we expected to be common among all the systems to be integrated and then to find which were necessary to complete the picture for systems with connectivity intelligence. Then each of these identified types was defined as an abstract entity. The information content is expressed through the interrelationship among all the entities. This interrelationship is known as the schema, which we codified through the EXPRESS language. The geometric entities we included were:

- Cells
- Mosaics (Arrays)
- Cell Instances
- Geometric Primitives:
    - Paths
    - Rectangles
    - Polygons
    - Lines
    - Points

The connectivity entities were:

- Element-Pin pairs (elPin)
- Nets

4.2. Layout Information Model Implementation

Figure 1 shows four physical design CAD systems, and two ROSE implementations for two different information models: the one introduced in the previous section and one that implements the Initial Graphics Exchange Standard (IGES). IGES is a very general (3-dimensional) standard to represent graphical information and was quite cumbersome for our purposes [NCGA]. However, we found it necessary to work with, because systems such as Allegro 6.1 (PCB oriented tools) often accept IGES representation of graphical data as their input, while not providing many other acceptable means to reliably input data from external sources, nor a procedural language to read or write to its internal data. Allegro 6.1 is, of course, also capable of writing out its layout design data in IGES form.

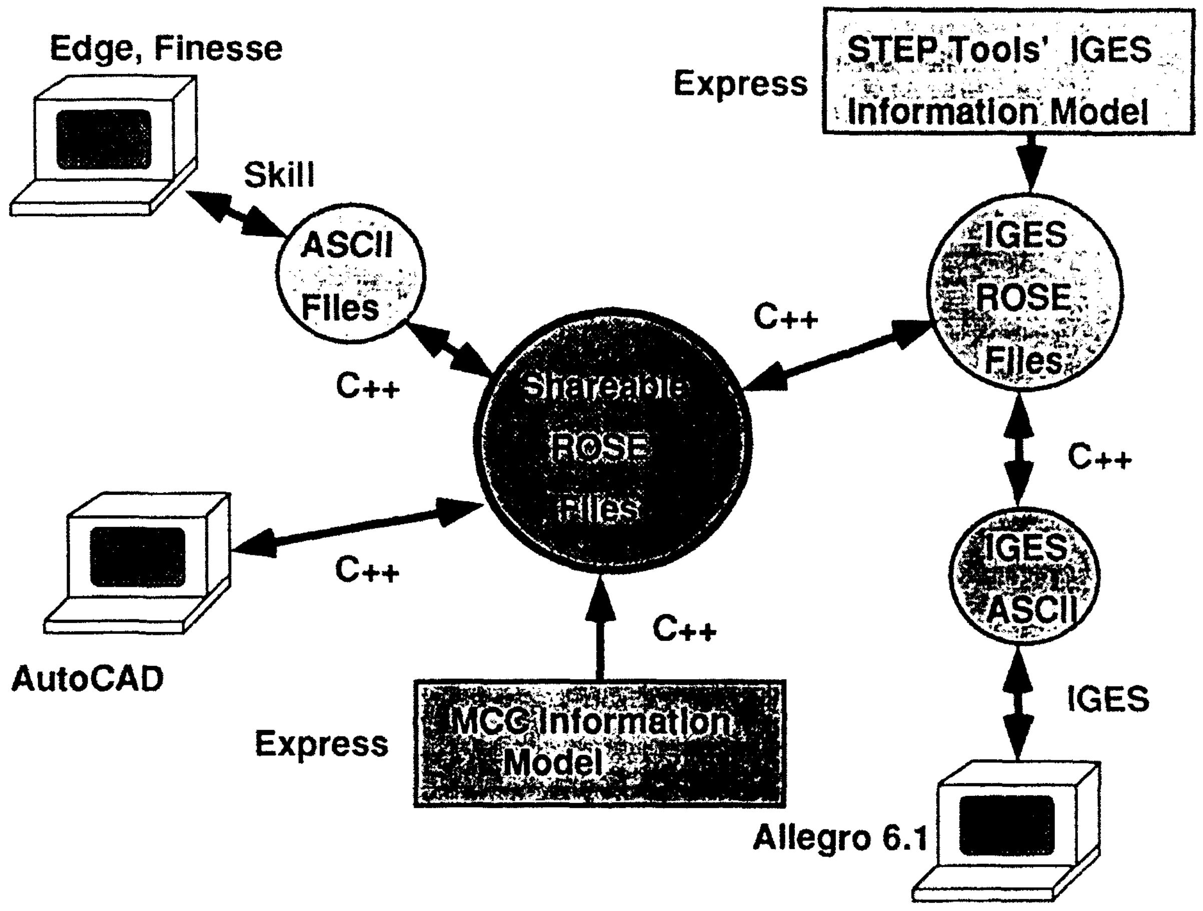


Figure 1. Architecture of the MOST system.

AutoCad is able to implement a more direct link between the MCC information model and its own internal format by making use of direct C procedural access to its database, and this provides a direct link for data exchange. AutoCad can read from and write to ROSE data directly, with no need for an intermediate format, simply by loading the C language interface that MCC has written using AutoCad's "AutoCad Development System" (ADS) tools.

The two information models, MCC's and IGES, were written in the EXPRESS language. Software tools from STEP Tools, Inc. were then used to compile those models into C++ classes. ROSE also provides methods to create the class instances, to relate them according to the schema, and to manage them. In addition, it was possible to map the ROSE objects created under one model to objects created under the other through C++ ROSE code, thereby integrating Allegro into MOST.

The link between Edge and ROSE is achieved through an intermediate text form. Data goes in and out of Edge via software written in Cadence's SKILL language, which allows read/write access to Edge's internal data structures. The link to ROSE was completed through a parser written in C++, which reads the intermediate form and creates the (C++) objects that implement the layout information model we introduced in the previous section. In the opposite direction, C++ software navigates through the ROSE objects and creates the intermediate form description, which is then read by the Cadence Skill code.

The Finesse system was linked to the ROSE database through an intermediate text file form called a "dfile", which is a standard textual database description form that Finesse supports, and which can be read back to recreate the Finesse database. Our original intention was to integrate Finesse in a direct fashion, without an intermediate text form, since Harris owns the source code for Finesse. However, it was determined by Harris Electronic Design Automation that such a route would take more time than that allowed by our project. This link was developed very much like the Cadence Edge link that was described in the previous paragraph.

4.3. Concurrent Engineering Layout System Operation

The operation of the system is visualized through Figure 2.

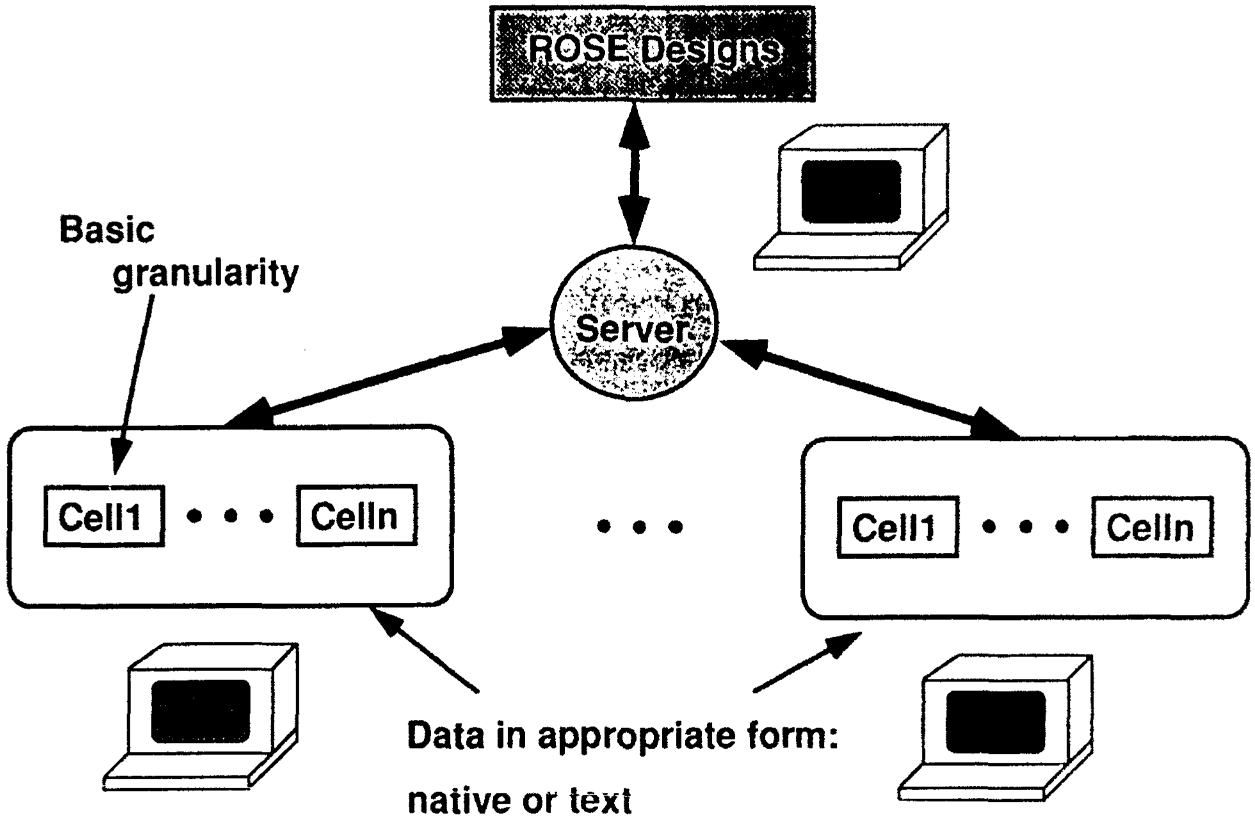


Figure 2. MOST system operation.

The basic object that is managed by ROSE is the Cell, and any transaction between a user and the system involves at least the transfer of a complete Cell entity. That implies that layout designers using the system should ensure that data is organized and structured in such a way as to make the transfer of data relatively painless by limiting the individual size of the cells they construct. Good data structuring is desired and a good practice regardless of concurrent engineering requirements, but in our case it is mandatory since N users of the system must be updated whenever one of them makes a change. It must be stressed here that ROSE in and of itself does not impose the restriction to have Cells as basic exchange entities. However, from a practical point of view, if one were to define the exchange granularity at a lower level, then there could be a prohibitively large amount of network traffic due to the vast number of minute changes that the design team may make. This would trigger corresponding ROSE database searches for the objects being modified, added or deleted. Since the natural unit of work in layout is the Cell, it is equally natural to use it as the transaction unit, and to have the ROSE data updated only when the designer finishes a sizable amount of work. This approach also eliminates the need for database searches for individual objects, as the whole cell gets updated every time. However, one must remember the need to keep the cells conveniently small.

Every Cell entity is stored as an individual ROSE design. The storage is handled by one dedicated workstation with efficient disk access. A custom server is invoked by the user to update the other workstations in the network

As a user receives an update, a backup copy of his current working design is kept. If a team member finds the update unacceptable for any reason, he must communicate with the other team members as to why that particular update is not good. After discussion and consensus is reached, either he will accept the update and modify whatever else he needs to make it palatable, or else the unacceptable cell will be further modified to meet his requirements. In any case, any designer with write privileges can always reinstate older backup cells back to ROSE, if that is the consensus.

4.4. Mechanism to save and update information

As outlined above, the Cell is the basic transaction unit we envision in our MCM layout concurrent engineering environment. As a designer completes work on a Cell, he saves it to his CAD system database and also invokes the ROSE server, who will receive the data and create the ROSE description of the Cell. Because the server nuns in a different workstation than that of the user, the designer will experience a minimum overhead from his saving to the ROSE environment. Since the Cell updates are not occurring frequently the network traffic is minimized. Once the Cell has been stored as a ROSE design, the ROSE server will undertake the task to translate it to all the other formats necessary to update the other CAD systems.

4.5. Special considerations

Certain CAD systems have implemented their internal databases with a minimum of hierarchy. Essentially, these systems, which have evolved from Printed-Circuit Board layout applications to the MCM domain keep "packages", "pad stacks" and "boards" as basic entities. A board describes the MCM, and contains packages and interconnect . The packages are built out of pad stacks and other geometric and electrical information. These systems are usually very useful to carry out the routing portion of the MCM layout task, but they are not especially suited to handle large amounts of data, notably for the case of certain MCM technologies which customize the MCM substrate starting from a generic, prefabricated part which is personalized in the last stage of manufacturing [DC91]. Such parts are mass produced ahead of time and contain a great deal of interconnect which will end up not used by the personalization step (typical utilization runs up to about 60% of available interconnect density for those substrates). In such cases, it is better to describe the part in full in a hierarchical CAD system. If one wants to use a non hierarchical CAD system for the routing in a concurrent engineering approach it is convenient to pass only the placement information and the netlist, and then to store back into ROSE the interconnect generated. Alternatively, if it proves to be possible to work effectively in the non-hierarchical system with all the MCM data, it would not be advisable to store back into ROSE the full design after routing since that would create a flat and large Cell with all the layout data in it. In any case what one needs is to only store the result of the interconnect (routing) step in a new Cell, to be shared by all the Concurrent Engineering Environment users. Furthermore, if it is possible to carry out the interconnect step itself in a sequence of smaller interconnect tasks (e.g. memory subsystem only, followed by others) it is advisable to store the results of each of those tasks as independent Cells.

Yet another scenario, which is not generally applicable but which covers a large portion of cases, consists in having all members of the design team collaborate in the definition of the basic packages and their placement, and then to submit the interconnect task to the non-hierarchical system (or systems) as the last step of the layout design activity.

4.6. UNIX server

In order to integrate the users of the concurrent engineering environment that we are creating we have implemented a mechanism by which the UNIX system will automatically update and distribute the changes to the ROSE database. Essentially, every user has his/her usual working environment, but in addition has two extra directories: One, labeled incoming and the other outgoing. If the user wants to update the ROSE database, he saves (using his CAD's system save command) to the outgoing directory (in addition to wherever he usually saves his CAD's system database). Once the server is invoked, the directory is "moved" or renamed to a predetermined location which the UNIX server periodically inspects. If the server finds data there, it makes a backup copy and it performs all necessary manipulations to change the new data into the MOST form and from there translates it to all other integrated systems. This frees the user and his workstation from that overhead, as the server normally runs independently in another machine, with enough memory, disk and CPU power, typically a SPARCStation 2 with 50+ Megabytes of memory and, depending on the size of the database an number of designs being managed, enough storage space.

Once the translations are complete, the server copies the ROSE database to an official directory where it is kept, and a backup copy of the previous data is made. The translations are distributed by the server to all registered users according to their recorded CAD system type, and copies placed in their "incoming" directories. Users are registered in a special file that the server reads when it is initially invoked. The users periodically inspect their incoming directories and can update their designs with the new distributed information. After they accept the changes, they can "move" their incoming directory to another location, and are ready to receive any new updates.

If they don't accept or agree with the changes, they need to communicate with the Concurrent design team members, discuss why the change is not acceptable and come to a consensus. If the change does stand, nothing is done. If the change is not accepted, then the old data is still in backup, or else the user that initiated the criticism on the change can submit his own data again to the server, and the change will be superseded.

The server mechanism was implemented in this fashion as an interim solution. It would be useful to have ROSE implement both a database version control and a design checking/locking mechanism. Once these features are available, the server can invoke them on behalf of the user that is submitting the change or requesting a specific piece of information or version of the data.

4.7. DDR2 parts and netlist

The module we designed concurrently utilizing the MOST system is known as the DDR2 (Digital Drop Receiver, version 2) and is proprietary to Harris Corp. The parts were designed using Finesse and the netlist had been entered into the Finesse system. From there , that information was made available to the other CAD systems through the ROSE database and the MOST system.

4.8. Installing and supporting the system at Harris Corporation

The system was installed at Harris Corporation in the month of October, 1992. A team of four software developers, three from MCC and one from Harris' Electronic Automation Division (Harris EDA) traveled to Florida and installed the system. A demonstration was given to Harris' designers, which consisted of translating the DDR2 placement and netlist from Harris EDA's Finesse to the Allegro 6.1 system. Those two CAD systems, Allegro and Finesse were the relevant design systems at Harris Corporation.

After the demonstration, Harris personnel were interested in trying an application involving a translation of the design from Finesse to Allegro, but utilizing a different technology, that used by the General Electric High Density Interconnect. The fully routed design existed as a Finesse database and it was necessary to translate it into an Allegro database. We used the same approach as the one we demonstrated. Up to that point we had not sent a fully routed database out of Finesse into the other design systems. We only had experience in sending fully routed designs out of Allegro and sharing them with the other CAD systems.

That specific application uncovered a few problems that had not been observed up until then. As we have reported previously (PR1, PR2, PR3) the path to go from the ROSE database to the Allegro system takes three steps: a) Map the MCC ROSE objects into IGES ROSE objects, b)Translate the IGES ROSE objects into an IGES file and c) Read the IGES file into Allegro. Step a) is accomplished using MCC's developed interface. Step b) utilizes STEP Tools' developed software, and step c) utilizes Cadence/Valid proprietary software. The observed problems were: 1) Step a) took several hours and 2) Step c) was taking days.

These problems were surprising as in our experience moderately large designs did not take that long to translate. We used UNIX's profiling utility to determine which part of MCC's code was executing most, and after that determination it was very easy to trace the problem to a simple bug in the code that did not alter the accuracy of the program but was executing redundant instructions. The code was straightforwardly debugged and the total execution time (elapsed time) of step a) went from 5 hours to 10 minutes on a Sun SPARCStation 2.

Resolving the bottleneck in step c) was more difficult because MCC does not own the source code. It was determined that there were the following circumstances which needed to be avoided in order to accelerate the time for step c): 1) Allegro uses a "blank board" as the basis on which it builds the design out of the IGES file. The board needed to have the netlist read in as a preliminary step. 2) Normally, as the Allegro database is created, Design Rule and Connectivity checking takes place, and that consumes time. In order to accelerate the data creation process, it is possible and desirable to turn off those checks. After we took those steps, the total time in taking the design from Finesse to Allegro was 4 hours (elapsed time) of which 30 minutes were ROSE related manipulations, and the balance strictly Cadence's IGES input into Allegro. This is for a fully routed, complex real design. The obtained Allegro database for chip pins, MCM I/O pads and signal interconnect is shown in Figure 3.

It turns out that Allegro does not take nearly as long to write its database in IGES file form as it takes to read it in. The same design that needs 3.5 hours to get in, Allegro can write out in just a few minutes.

4.9. Concurrent Engineering Design Experience at MCC

In order to validate the concurrent engineering design environment provided by the MOST system, MCC carried out a design of the DDR2 Harris module. We took the parts description, placement and netlist from the database that Harris delivered in Finesse form. The ROSE database was created and the Cadence EDGE system read it in. The footprints of the chips and the module I/O were then modified to allow for automatic redundant routing, by judicious modification of the padstack definitions. The placement data and the netlist were then read by the Allegro system and routed. The Allegro system also created the power and ground plane data. The routed Allegro database was then sent back to the ROSE database and read in by the Cadence EDGE system, which was then used to replicate the signal and power/ground layers. The via entities were also redefined in the Cadence EDGE system to have geometric elements in the normal as well as the redundant routing layers. The replication step is trivial in the EDGE system but not in the Allegro or Finesse systems. MCC has also written special verification software aids to take the netlist, chip placement and netlist information residing in the ROSE database and then generate a text file whose records list the net names followed by the absolute x and y Cartesian coordinate location of the chip and module I/O pins corresponding to the listed net name. Such a text file is read into the Cadence Edge database using the SKILL language and the corresponding net names written as labels into the native Cadence database in a chosen layer at the x,y location recorded in the text file. Such annotation is the basis on which the Edge connectivity verification software runs. Through simple textual editing operations on the text file it is possible to derive a corresponding text file for the redundant net names, which are then also input as labels into the Edge database (in a different layer than that used for the normal net names). Through straightforward graphical operations the labels are moved to desired locations on the redundant pads , then moved to the same layer as the normal net name labels. At that point the Edge database is ready for full connectivity and design rule verification.

It is important to stress that the method we have just described is exceedingly powerful as it allowed us to perform design operations that normally are not allowed in connectivity-based systems, but which are very desirable. Those "forbidden" operations, such as adding layers, replicating design data, adding connections not listed in the original netlist but which are necessary to easily implement redundancy are practically trivial in the EDGE system. Through the tool integration and full verification of the final design we are assured of the integrity of the design.

In our experiment, the total designer time spent was 4 man days and we obtained a fully verified database, suitable for manufacturing. The design rules utilized were those of MCC's QTAI thin film MCM interconnect process [PR1, DC91]. In comparison, the same design when processed through the standard non-concurrent approach takes one man month of effort. The savings in time are mainly due to: 1) The ability to use and re-use previously designed data elements, in our case the initial Finesse database already contained the chip and module I/O footprint definitions. The reusability of data is a powerful capability that eliminates transcription errors and saves a great deal of design time. 2) The ability to use the best features of each and every integrated tool.

4.10. ARPA demonstrations

The system was first demonstrated at ARPA headquarters to several officers and interested persons. One notable fact was that there was an engineering group from a large systems company that attended one of the demonstrations and manifested interest in the environment as a basis to establish an interface between design and manufacturing.

The system was demonstrated a second time at the DICE Phase 5 kick-off meeting in Arlington, VA for DICE contractors and other interested parties.

5. Technical Results

5.1 Cadence EDGE - MOST link established

The first principal tèchnical result was that a bi-directional link was established and demonstrated between the Cadence Edge and the ROSE systems. The fundamentals of how this link works have been described above and in particular in the discussion of Figure 1. The MCC information model was written as a schema in the EXPRESS language. This schema was then compiled into C++ classes through the express2c++ software tool from STEP Tools, Inc.

At the same time, an interface to the internal database of Cadence Edge was developed, making use of the SKILL language from Cadence Design Systems. The role of that interface is to allow the user of Edge to save his work both to the Cadence data format and also into ROSE objects, and also to allow him to read in any updates. The mechanism, as d^scribed earlier is to go through an intermediate form. That form is picked up by the other side of the ·ftware interface we built. On the way in to the ROSE system, the intermediate form is parsed, on the way out of ROSE the object representation of the Cells are written in the intermediate form.

It should be noted that when the Cadence Edge system reconstructs the Cells from the intermediate form, it requires the procedure to occur from the bottom up, that is, simple Cells must be constructed before building more complex ones. Since every Cell is represented by an individual intermediate file and an individual ROSE design file, it is necessary to process the files in the right order. A special auxiliary file is created which lists the subcells of any given cell in the right order.

The layout schema we have utilized is included in this report as Appendix A.

The parser of the intermediate form into ROSE objects was written using YACC++. A simple lexical analyzer was written in this context. The code to write out ROSE objects into the intermediate form was written in C++ and relies heavily in the ROSE methods to navigate and manage the objects.

5.2. Cadence Edge, Allegro, Finesse and AutoCad systems integrated

A bi-directional link was established and demonstrated between each of the Cadence Edge, Allegro, Finesse CAD systems and the MOST central database The fundamentals of how these links work have been described above (see Section 4 of this report). The MCC information model was further improved at this stage, by eliminating some entities such as Donuts and Ellipses which have no universal use among all CAD systems.

A further technical result in the software area was the development of an AutoCad to MOST interface. All of the entities in the previously reported (PR1, PR2, PR3) schema have been implemented. This interface allows the inexpensive design of parts, chip footprints, module I/O, etc. and the sharing of this data with all the other CAD systems. A special virtue of this interface is that it has been developed in C++ and it is a direct interface between the AutoCad database and the ROSE database, which does not necessitate an intermediate text file.

An additional benefit is derived from the fact that AutoCad can be run on a relatively inexpensive personal computer, thereby reducing the cost of adding CAD seats to a design team. This advantage is, however, tempered by the physical limitations of a personal computer. AutoCad performs admirably when used to design even complex cells, but its performance decreases when handling designs containing large numbers of cell placements. Hence, it appears that until the manufacturers improve upon the method by which they handle external references, AutoCad's main usefulness will remain in the area of cell or component design.

Images taken from the CAD systems are included in this report to illustrate the capabilities obtained (see Figures 4 through 9).

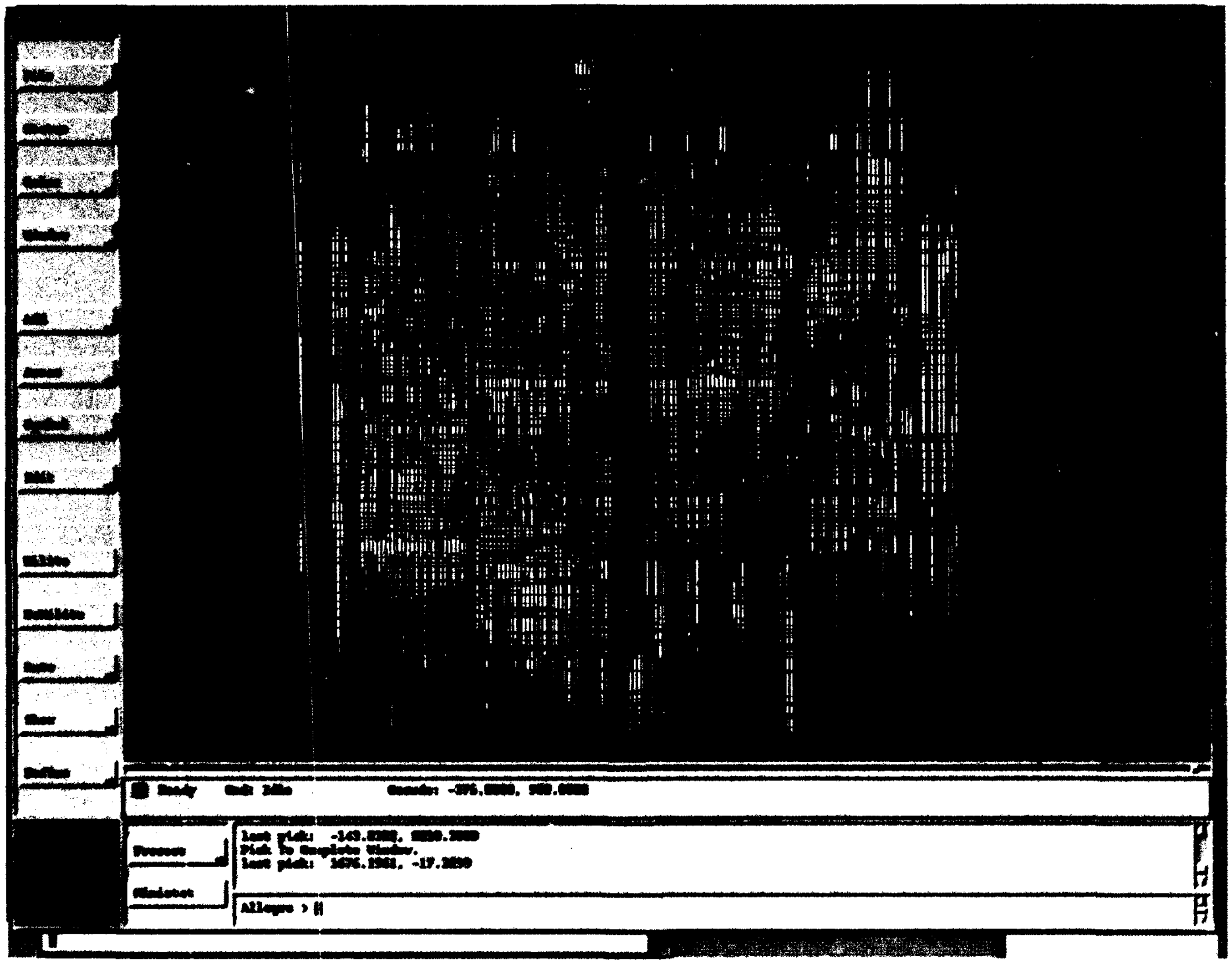
5.3. Initial Concurrent Design Experiment

Figure 4 shows a screen dump of a Sun Workstation where the initial data for the DDR2 module was entered. The system is Finesse, and the parts were created there and the netlist read into that system. Using our Finesse-ROSE interface, the initial database was created. It was transferred to both Cadence Edge and Allegro systems.

Figure 5 shows the design in the Allegro design system, and Figure 6 the initial design in the Cadence Edge system. After being extracted from the ROSE database in text form, the netlist was read into the Allegro system through a NETIN command which was fed to a blank Multi-Chip Module design. The physical design data for the DDR2 parts placement is then automatically created in that environment.

5.4. Concurrent Design Experiment

As we explained in section 4.9, MCC designed the DDR2 MCM using the QTAI design rules and implementing a redundant routing strategy. Figures 8 and 9 show the routed design in Allegro and Edge systems, respectively.



```
Ready    [illegible]    [illegible]: [illegible]

last pick:  -143.[illegible], [illegible]
Pick to [illegible] Window.
last pick:  [illegible], -17.[illegible]

Allegro > 
```

Figure 3: The Harris HDI DDR2 design after conversion from Finesse to the Allegro system.

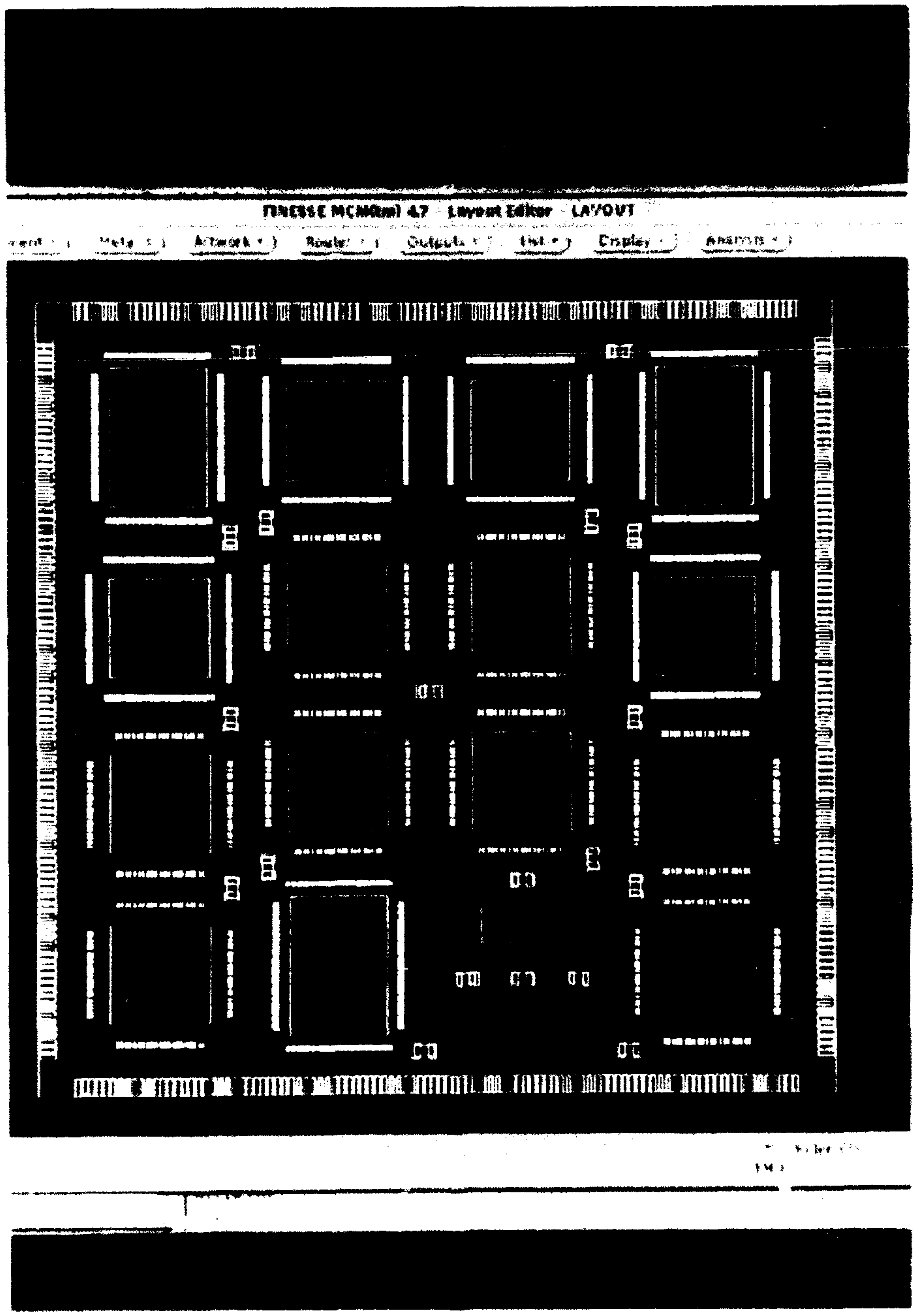
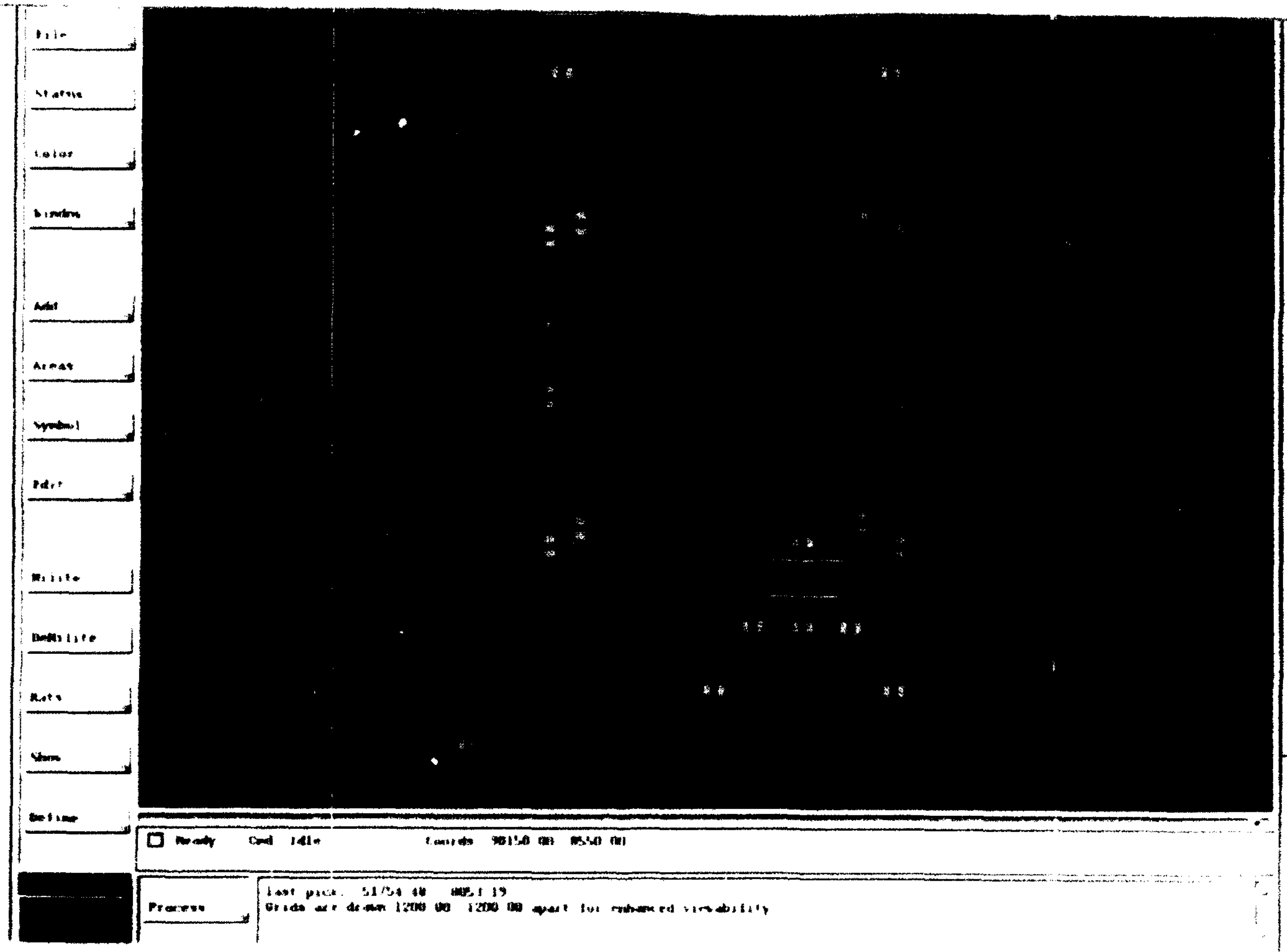


Figure 4: The unrouted Harris DDR2 design in the Finesse system.



```
Coords  98150 00  8550 00
Last pick.  51754 40  8051 19
Grids are drawn 1200 00  1200 00 apart for enhanced viewability
```

Figure 5: The unrouted DDR2 design converted to Allegro.

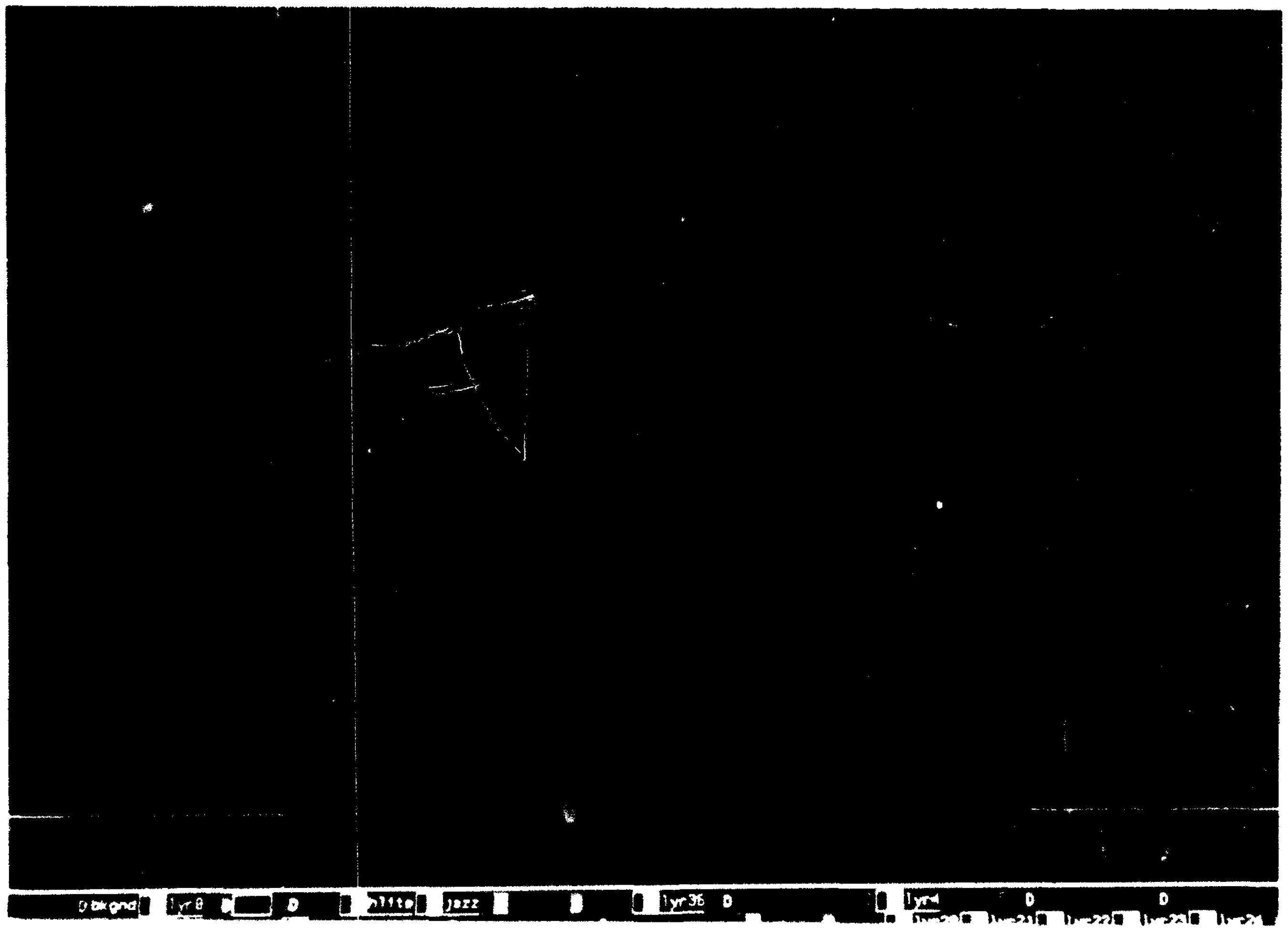


Figure 6: The unrouted DDR2 design converted to Cadence Edge.

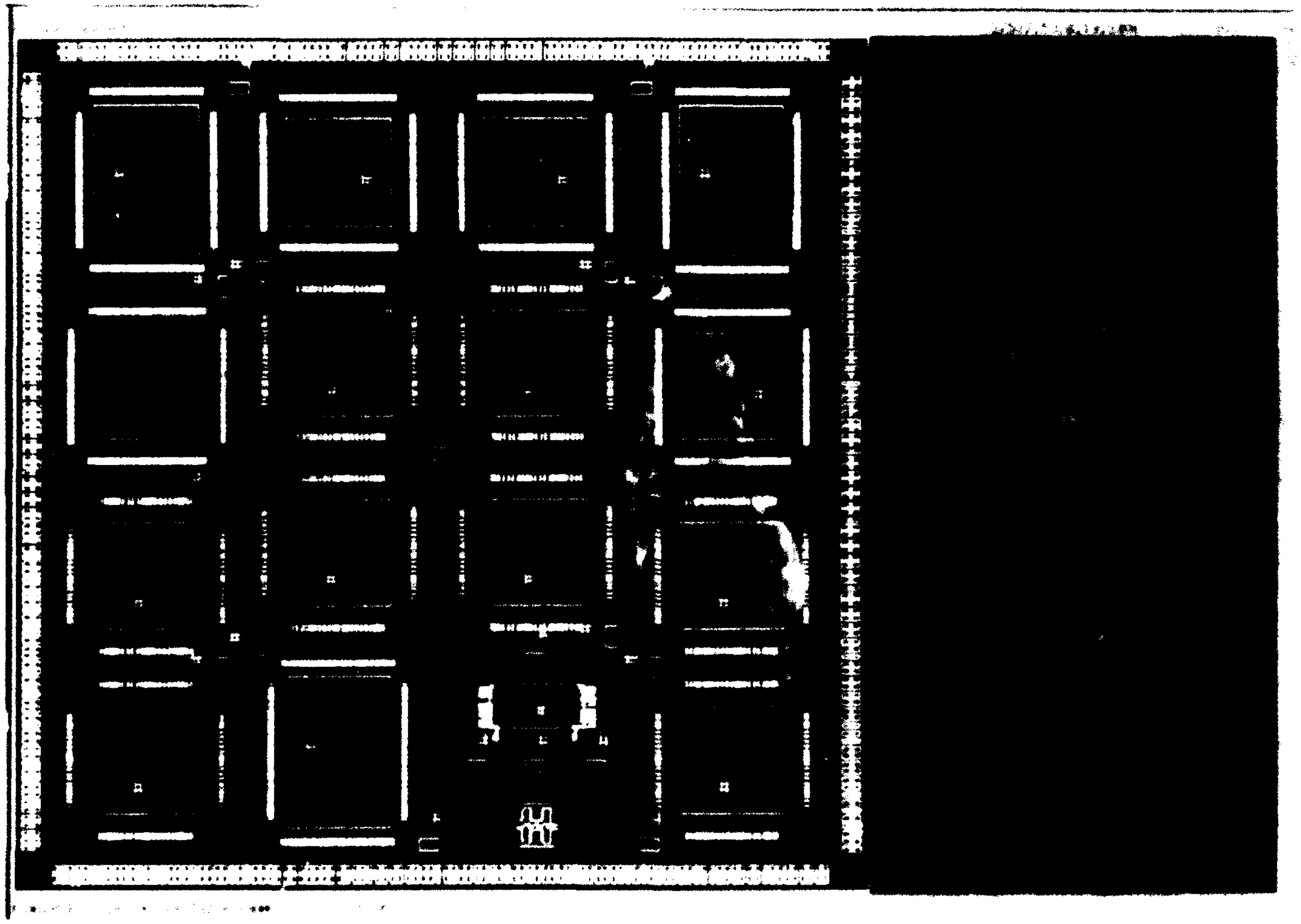
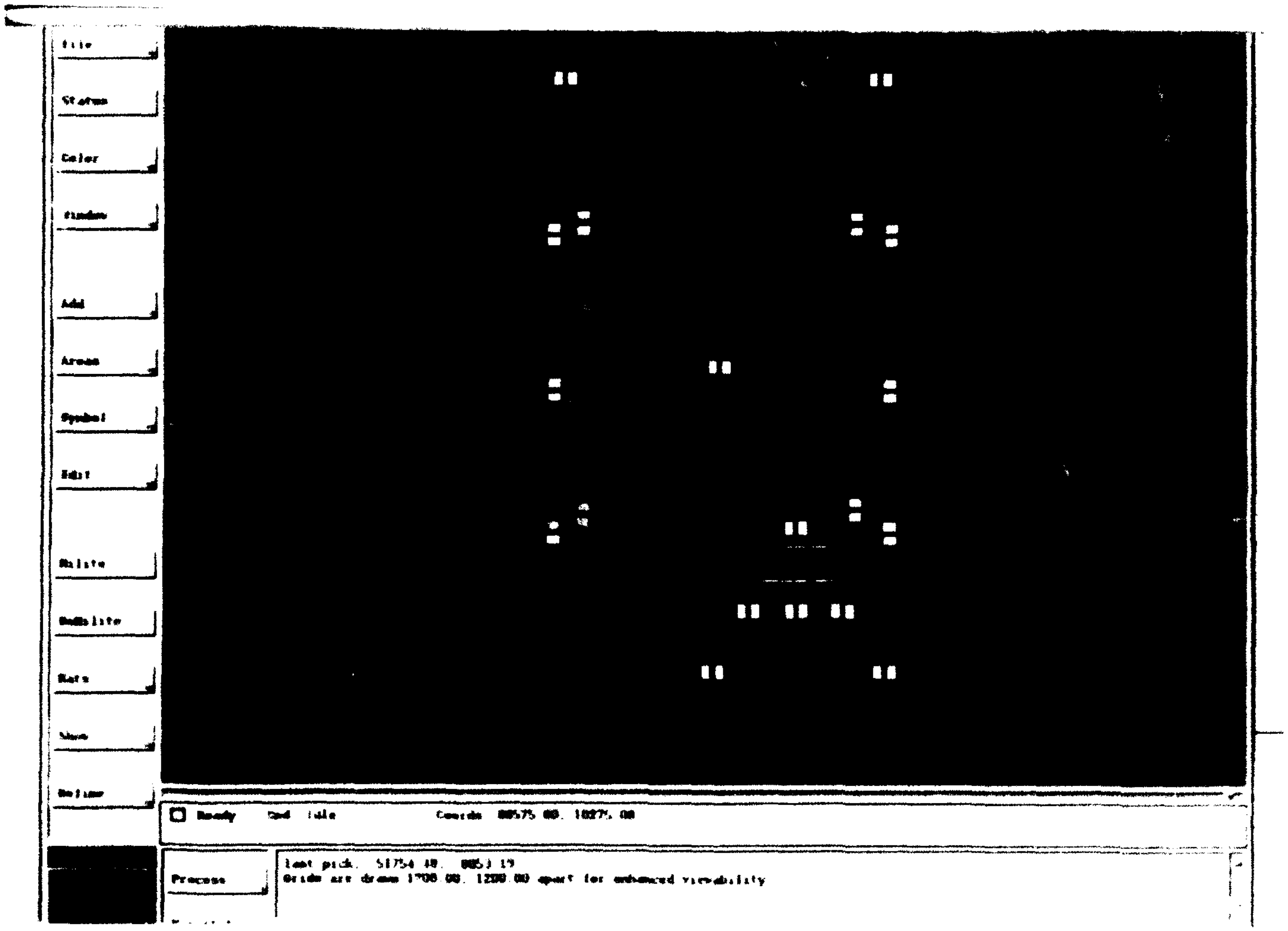


Figure 7 The unrouted DDR2 design converted to AutoCad



```
Coords 08575.00, 10275.00
Last pick: 51754.40, 0853.19
Grids are drawn 1700.00, 1200.00 apart for enhanced viewability
```

Figure 8: The DDR2 design after routing in Allegro

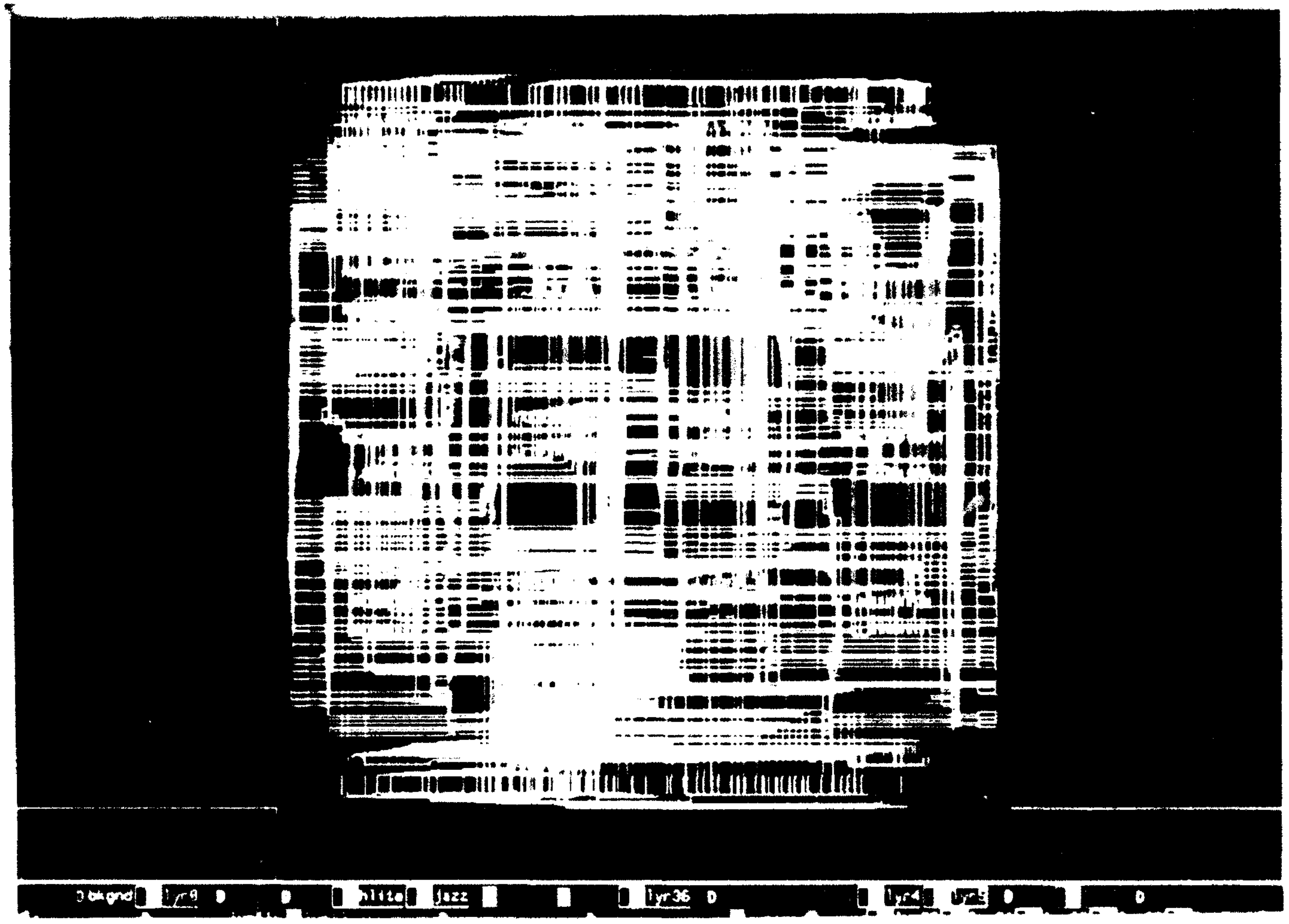


Figure 9: The singly routed DDR2 design after conversion to the Cadence Edge system where the routing redundancy was implemented.

6. Conclusions

We have successfully integrated the four MCM layout systems to the ROSE data manager, as required by the contract, and created a demonstration quality concurrent engineering environment that has been proven to be suitable for the design of MCMs at MCC. We have also established a strategy and a methodology suitable for the concurrent design effort, which was validated during the design activity portion of this project. It is important to note that the aforementioned system is currently at a demonstration level of functionality only, and that much work remains to be done in order to bring the system up to "industrial strength" and to make it suitable for future commercialization. However, it is clear to those involved with the development and use of this system that this paradigm, when implemented properly, can not only provide a significant increase in the speed of design activity (either by groups or individuals), but also increases the capabilities of the design team by allowing them to combine the best and most desirable features of each of the integrated CAD systems into a single, highly efficient, and flexible design tool.

References

[DC91] D. Carey et. al. "Variations in MCM Implementations Using a Semi-Custom Technology", Proceedings of the International Electronics Packaging Society (IEPS). 1991, pp. 94-106

[NCGA] IGES documentation can be obtained from the National Computer Graphics Association, IGES/PDES Organization, 2722 Merrilee Dr, Suite 200, Fairfax, VA 22031, (703)698-9600

[ST] "Tutorial Manual for the ROSE++ data manager" and "Reference Manual for the ROSE++ data manager", STEP Tools Inc., Rensselaer Technology Park, Troy, NY, 12180, (518) 276-6751

[PR1] Hector Moreno, An MCM/Chip Concurrent Engineering Validation, 1992 Second Quarter Technical Report, MCC Report # HVE-171-92 (1992).

[PR2] Hector Moreno, An MCM/Chip Concurrent Engineering Validation, 1992 Third Quarter Technical Report, MCC Report # HVE-232-92 (1992).

[PR3] Hector Moreno, An MCM/Chip Concurrent Engineering Validation, 1992 Fourth Quarter Technical Report, MCC Report # HVE-010-93 (1993).

Appendix A
Layout Information Model, V2.0

```
(*****************************************************************************)
(*                                                                           *)
(* Copyright 1992, Microelectronics and Computer Technology Corporation     *)
(*                    All rights reserved                                    *)
(*                                                                           *)
(*****************************************************************************)
SCHEMA layout;


ENTITY Property;
        name: STRING;
        value: STRING;
END_ENTITY;


ENTITY Layer
        SUBTYPE OF (LayoutObj);
        lyr: INTEGER;
END_ENTITY;


ENTITY Point
        SUBTYPE OF (LayoutObj);
        x: REAL;
        y: REAL;
END_ENTITY;


ENTITY BBx
        SUBTYPE OF (LayoutObj);
        ll: Point;
        ur: Point;
END_ENTITY;


ENTITY Line
        SUBTYPE OF (LayoutObj);
        lyr: Layer;
        nPath: INTEGER;
        path: LIST OF Point;
END_ENTITY;


ENTITY Path
        SUBTYPE OF (LayoutObj);
        beginExt: REAL;
        endExt: REAL;
        lyr: Layer;
        netNum: INTEGER;
        nPath: INTEGER;
        path: LIST OF Point;
        pathShape: STRING;
        width: REAL;
END_ENTITY;


ENTITY Rectangle
        SUBTYPE OF (LayoutObj);
        bBox: BBx;
        lyr: Layer;
END_ENTITY;


ENTITY Polygon
        SUBTYPE OF (LayoutObj);
        lyr: Layer;
        nPath: INTEGER;
        path: LIST OF Point;
END_ENTITY;


ENTITY Circle
        SUBTYPE OF (LayoutObj);
        center: Point;
        radius: REAL;
        lyr: Layer;
END_ENTITY;


ENTITY Label
        SUBTYPE OF (LayoutObj);
        draftingP: BOOLEAN;
        font: STRING;
        height: REAL;
        justify: STRING;
        labelType: STRING;
        lyr: Layer;
        orient: STRING;
        angle: REAL;
        theLabel: STRING;
        xy: Point;
END_ENTITY;


ENTITY Cell
        SUBTYPE OF (LayoutObj);
        blockName: STRING;
        cellType: STRING;
        objList: LIST OF LayoutObj;
END_ENTITY;


ENTITY Instance
        SUBTYPE OF (LayoutObj);
        master: STRING;
        blockRef: STRING;
        orient: STRING;
        xy: Point;
END_ENTITY;


ENTITY Mosaic
        SUBTYPE OF (LayoutObj);
        columns: INTEGER;
        rows: INTEGER;
        master: STRING;
        uX: REAL;
        uY: REAL;
        xy: Point;
        orient: STRING;
END_ENTITY;


ENTITY ElPin
        SUBTYPE OF (LayoutObj);
        pinName: STRING;
        pinNum: INTEGER;
        padRef: STRING;
        pkgRef: STRING;
END_ENTITY;


ENTITY Net
        SUBTYPE OF (LayoutObj);
        netName: STRING;
        netNum: INTEGER;
        elPinList: LIST OF ElPin;
END_ENTITY;


ENTITY LayoutObj
        SUPERTYPE OF (ONEOF (Point, Layer, Circle, Line, Path, Polygon,
                        Rectangle, Label, Cell, Instance, ElPin, Net));
        selfIdent: STRING;
        prop: LIST OF Property;
END_ENTITY;


END_SCHEMA;
```

Acknowledgments

Team Members

Chuck Gannon from Harris Electric Design Automation has implemented the Finesse - ROSE interface. He participated in the Harris and ARPA demonstrations.

Lou Paradiso, Charles Davies and Lee Shombert from Harris Government and AeroSpace Division participated in the exercise of the software and provided the DDR2 module definition.

Deborah Cobb from MCC has collaborated with her expertise with the Allegro system and its IGES characteristics. She has also collaborated in extracting device characteristics out of the ROSE database for the Allegro NETIN command. She proposed the layer replication method for redundant routing used in MCC's concurrent design experiment and routed the design in Allegro. She participated in the Harris Corp. demonstration of the system.

Carroll Vance from MCC has implemented netlist extraction and creation for the Allegro system, as well as the label extraction from the ROSE database for purposes of design verification on the EDGE system.

Christopher Hudnall from MCC has been helpful with all matters related to the installation of ROSE at MCC, as well as the UNIX support and server implementation. He participated in both Harris and ARPA demonstrations of the system.

External Support

It is a pleasure to acknowledge the support of Martin Hardwick, Alok Mehta, Blair Downie (STEP Tool's IGES translation toolkit), David Loffredo and in general of the employees of STEP Tools, Inc., in all matters regarding ROSE.

Special thanks are due to Mike Gilliam and his team at ARPA's Information Resource Center for his help in all matters related to our system demonstration there.